Top-L Most Influential Community Detection Over Social Networks

Nan Zhang
East China Normal University
Shanghai, China
51255902058@stu.ecnu.edu.cn

Yutong Ye
East China Normal University
Shanghai, China
52205902007@stu.ecnu.edu.cn

Xiang Lian
Kent State University
Kent, United States
xlian@kent.edu

Mingsong Chen
East China Normal University
Shanghai, China
mschen@sei.ecnu.edu.cn

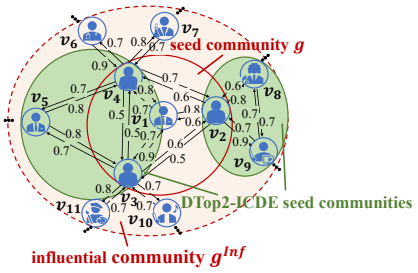
***Abstract*—In many real-world applications such as social network analysis and online marketing/advertising, *community detection* is a fundamental task to identify communities (subgraphs) in social networks with high structural cohesiveness. While previous works focus on detecting communities alone, they do not consider the collective influences of users in these communities on other user nodes in social networks. Inspired by this, in this paper, we investigate the influence propagation from some *seed communities* and their influential effects that result in the *influenced communities*. We propose a novel problem, named *Top-L most Influential Community DEtection* (Top$L$-ICDE) over social networks, which aims to retrieve top-L seed communities with the highest influences, having high structural cohesiveness, and containing user-specified query keywords. To efficiently tackle the Top$L$-ICDE problem, we design effective pruning strategies to filter out false alarms of seed communities and propose an effective index mechanism to facilitate efficient Top-L community retrieval. We develop an efficient Top$L$-ICDE answering algorithm by traversing the index and applying our proposed pruning strategies. We also formulate and tackle a variant of Top$L$-ICDE, named *diversified top-L most influential community detection* (DTop$L$-ICDE), which returns a set of L diversified communities with the highest diversity score (i.e., collaborative influences by L communities). We prove that DTop$L$-ICDE is NP-hard, and propose an efficient greedy algorithm with our designed diversity score pruning. Through extensive experiments, we verify the efficiency and effectiveness of our proposed Top$L$-ICDE and DTop$L$-ICDE approaches over real/synthetic social networks under various parameter settings.**

***Index Terms*—Top-L Most Influential Community Detection, Diversified Top-L Most Influential Community Detection**

I. Introduction

Recently, the *community detection* (CD) has gained significant attention as a fundamental task in various real-world applications, such as online marketing/advertising [1]–[3], friend recommendation [4], and social network analysis [5]. Many previous works [6]–[9] usually focused on identifying communities only (i.e., subgraphs) with high structural cohesiveness in social networks. However, they overlooked the collective influences that communities may exert on other users (e.g., family members, or friends) within social networks, which play a significant role in Word-Of-Mouth effects.

In this paper, we will formulate and tackle a novel problem called *top-L most influential community detection over social networks* (Top$L$-ICDE). This Top$L$-ICDE problem aims to detect top-L communities of people from social networks who have specific interests (e.g., sports, movies, traveling), not only with close social relationships (i.e., forming dense subgraphs with highly connected users) but also with high impacts/influences on other users in social networks.



(a) social network G

User	Keyword Set
v_1	{Movies, Books}
v_2	{Movies, Food}
v_3	{Movies, Jewelry}
v_4	{Movies, Crafts}
v_5	{Movies, Health}
v_6	{Health, Wellness}
v_7	{Books, Movies}
v_8	{Home Decor, Movies}
v_9	{Movies, Books}
v_{10}	{Cosmetics, Skincare}
v_{11}	{Jewelry, Movies}

(b) keyword sets of vertices

Fig. 1. An example of the Top$L$-ICDE problem over social network.

Below, we give a motivating example of our Top$L$-ICDE problem in real applications of online advertising/marketing.

Example 1 ***(Online Advertising and Marketing over Social Networks)*** *In real applications of online advertising and marketing, a sales manager wants to find several communities of users on social networks who might be interested in buying a new product. Intuitively, people in the same communities who know each other should share a sufficient number of common friends within communities, so that they are more likely to purchase the product together via group buying (e.g., on Groupon [10]). What is more, after such community users receive group buying coupons/discounts, the sales manager wants these users to maximally influence other users on social networks via their posts/recommendations. In this case, the sales manager needs to perform the Top$L$-ICDE operator to find some seed communities with the highest influences on social networks (also considering those influenced users).*

Figure 1 illustrates an example of a social network, G, which contains user vertices like $v_1 \sim v_5$, and directed edges (e.g., edge e_{v_1,v_2}) representing the relationships (e.g., friend, colleague, or family) between two users (e.g., v_1 and v_2). As illustrated in Figure 1(a), each edge has two influence weights between two vertices, for example, directed edge e_{v_1,v_2} has the weight 0.8, indicating the influence from user v_1 to user v_2. Moreover, as depicted in Figure 1(b), each user is associated with a set of keywords that represent one's favorite product categories (e.g., $\{Movies, Books\}$ for user v_1).

To achieve good online advertising/marketing effects, one potential seed community, g, is shown in Figure 1(a) (i.e., inner solid circle), where users in g are all interested in "Movies",

with high structural connectivity (i.e., any two users in g are friends and also share two other common friends), and have the highest influences on other user nodes in a larger influenced community g^{Inf} (i.e., outer dashed ellipse, as shown in Figure 1(a)). ■

Inspired by the example above, in this paper, we consider the Top$L$-ICDE problem, which obtains L groups of highly connected users (called *seed communities*) containing some query keywords (e.g., shopping preferences or preferred products) and with the highest influences (ranks) on other users. The resulting L communities are potential customer groups for effective advertising/marketing with high influences to promote new products in social networks.

Formally, we will first define a seed community g in social networks as a structurally dense subgraph (under the semantics of (k, r)-truss [11], [12]), where any two connected users in g must have at least $(k-2)$ common friends (i.e., be contained in $\geq (k-2)$ triangle motifs, indicating stable user relationships), users in g are close to a center vertex (i.e., within r hops), and each individual vertex in g contains at least one query keyword. Then, we will consider the influenced communities under an influence propagation model from seed communities, which are ranked by influential scores of seed communities. Our Top$L$-ICDE problem aims to retrieve top-L seed communities with the highest ranks.

Due to the large scale of social networks and the complexities of retrieving communities under various constraints, it is rather challenging to efficiently and effectively tackle the Top$L$-ICDE problem over big social networks. Furthermore, since many parameters like query keywords or thresholds (e.g., influential score, radius, and support) are online specified by users, it is not clear how to enable efficient Top$L$-ICDE processing with ad-hoc constraints (or query predicates).

In order to efficiently tackle the Top$L$-ICDE problem, in this paper, we propose a two-phase general framework for Top$L$-ICDE processing, which consists of offline pre-computation and online Top$L$-ICDE processing phases. In particular, we will present effective pruning mechanisms (w.r.t query keywords, radius, edge support, and influential scores) to safely filter out false alarms of candidate seed communities and reduce the Top$L$-ICDE problem search space. Moreover, we will design an effective indexing mechanism to facilitate our proposed pruning strategies and develop an efficient Top$L$-ICDE processing algorithm via index traversal.

Furthermore, we also consider a variant of Top$L$-ICDE, namely *diversified top-L most influential community detection over social networks* (DTop$L$-ICDE), which retrieves a set of L diversified communities with the highest diversity score. Different from Top$L$-ICDE that returns L individual communities (each of which can be a candidate community for online marketing/advertising), DTop$L$-ICDE obtains one set of L diversified communities that *collaboratively* influence other users with the highest diversity score (i.e., considering the overlaps of their influenced communities).

In Example 1 (Figure 1), if a sales manager wants to enhance the promotion effect of advertising to L seed communities, we need to consider those users influenced by more than one seed community. Since users usually buy the product once only, we would like to reduce the overlaps of L influenced communities and maximize the influential effect (i.e., the diversity score). In this case, we can issue the DTop$L$-ICDE variant to obtain L diversified communities. In Figure 1(a), for $L = 2$, subgraphs $\{v_3, v_4, v_5\}$ and $\{v_2, v_8, v_9\}$ are two DTop2-ICDE seed communities (with only one influenced user v_1).

We prove that the DTop$L$-ICDE problem is NP-hard, and develop an approximation greedy algorithm to process the DTop$L$-ICDE query efficiently.

We make the following major contributions in this paper:

1) We formally define the problem of the top-L most influential community detection over social networks (Top$L$-ICDE) and its variant DTop$L$-ICDE in Section II.
2) We design a two-phase framework to efficiently tackle the Top$L$-ICDE problem in Section III.
3) We propose effective pruning strategies to reduce the Top$L$-ICDE search space in Section IV.
4) We devise offline pre-computation and indexing mechanisms in Section V to facilitate effective pruning and an efficient Top$L$-ICDE algorithm to retrieve community answers in Section VI.
5) We prove that DTop$L$-ICDE is NP-hard, and develop an efficient DTop$L$-ICDE processing algorithm to retrieve diversified community answers in Section VII.
6) We demonstrate through extensive experiments the efficiency and effectiveness of our Top$L$-ICDE processing approach over real/synthetic graphs in Section VIII.

Section IX reviews previous works on community search/detection, influence maximization, and influential / diversified community. Finally, Section X concludes this paper.

TABLE I
SYMBOLS AND DESCRIPTIONS

Symbol	Description
G	a social network
$V(G)$	a set of n vertices v_i
$E(G)$	a set of edges $e(u, v)$
g (or g_i)	a seed community (subgraph)
$p_{u,v}$	a propagation probability that user u activates its neighbor v
g^{Inf} (or g_i^{Inf})	the influenced community of a seed community g (or g_i)
$hop(v_i, r)$	a subgraph centered at vertex v_i and with radius r
$v_i.W$	a set of keywords that user v_i is interested in
$v_i.BV$	a bit vector with the hashed keywords in $v_i.W$
$\sigma(g)$	the influential score of the influenced community g^{Inf}
Q	a set of query keywords
$Q.BV$	a bit vector with the hashed query keywords in Q
k	the support threshold in k-truss for seed communities
$sup(e_{u,v})$	the support of edge $e_{u,v}$
r_{max}	the maximum possible radius of seed communities
r	the user-specified radius of seed communities
θ	the influence threshold

II. PROBLEM DEFINITION

Section II-A formally defines the data model for social networks. Section II-B gives the definition of the information propagation model over social networks. Finally, Section II-C provides the definition of our problems.

A. Social Networks

First, we model a social network by an attributed, undirected, and weighted graph as follows.

Definition 1 *(**Social Network,** G) A social network G is a connected graph represented by a triple $(V(G), E(G), \Phi(G))$, where $V(G)$ and $E(G)$ are the sets of vertices and edges in G, respectively, and $\Phi(G)$ is a mapping function: $V(G) \times V(G) \rightarrow E(G)$. Each vertex v_i has a keyword set $v_i.W$, and each edge $e_{u,v} \in E(G)$ is associated with a weight $p_{u,v}$, which indicates the probability that user u activates user v.*

In Definition 1, the keyword set $v_i.W$, e.g., $\{movies, sports, \cdots\}$, denotes the topics that user v_i is interested in.

B. Information Propagation Model

To describe the influence spread over social networks G, we utilize the *maximum influence arborescence* (MIA) model [13]. Given a specific path from u to v (i.e., a non-cyclic user sequence), denoted as $P_{u,v} = \langle u = u_1, u_2, \ldots, u_m = v \rangle$, the *propagation probability*, $pp(P_{u,v})$, of path $P_{u,v}$ is given by:

$$pp(P_{u,v}) = \prod_{i=1}^{m-1} p_{u_i,u_{i+1}}. \tag{1}$$

where $p_{u_i,u_{i+1}}$ is the weight of edge $e_{u_i,u_{i+1}}$ on path $P_{u,v}$.

In Eq. (1), we give the probability of the influence propagation between two user vertices via a specific path $P_{u,v}$ in G. In reality, there are multiple possible paths from u to v. Thus, the MIA model uses the *maximum influence path* (MIP) [13], denoted as $MIP_{u,v}$, to evaluate the influence propagation from u to v. The MIP is defined as a path with the highest propagation probability below:

$$MIP_{u,v} = \underset{P_{u,v}}{argmax}\, pp(P_{u,v}). \tag{2}$$

This way, the *user-to-user propagation probability*, $upp(u, v)$, for all paths between users u and v is defined by:

$$upp(u, v) = pp(MIP_{u,v}). \tag{3}$$

The problem of computing the influence spread $\sigma(S)$ is known to be NP-hard [13], and existing algorithms [14], [15] can achieve an approximation factor of $(1 - 1/e + \varepsilon)$, where e is the natural constant and $\varepsilon > 0$.

C. Our TopL-ICDE Problem

Seed Community. We first give the definition of the *seed community* in social networks G below.

Definition 2 *(**Seed Community,** g) Given a social network G, a center vertex v_q, an integer support k, the maximum radius, r, of seed communities, and a set, Q, of query keywords, a seed community, g, is a connected subgraph of G (denoted as $g \subseteq G$), such that:*

- *$v_q \in V(g)$;*
- *for any vertex $v_l \in V(g)$, we have $dist(v_q, v_l) \leq r$;*
- *g is a k-truss [16], and;*
- *for any vertex $v_l \in V(g)$, its keyword set $v_l.W$ must contain at least one query keyword in Q (i.e., $v_l.W \cap Q \neq \emptyset$), where $dist(x, y)$ is the shortest path distance between x and y in subgraph g.*

The seed community g (given in Definition 2) is a connected subgraph that is 1) centered at v_q, 2) with a maximum radius r, 3) being a k-truss, and, 4) with each vertex v_l containing at least one query keyword in Q. Here, g is a k-truss subgraph [16], if any edge in g is contained in at least $(k-2)$ triangles.

In real applications of online marketing and advertising, the seed community usually contains strongly connected users, who are provided with coupons/discounts of products to maximally influence other users.

Influenced Community. The *influenced community*, g^{Inf}, is a subgraph influenced by a seed community g. Based on the MIA model, we define the *community-to-user propagation probability* from seed community g to a vertex v as:

$$cpp(g, v) = \begin{cases} \max_{\forall u \in V(g)}\{upp(u, v)\}, & v \notin V(g); \\ 1, & v \in V(g). \end{cases} \tag{4}$$

We use $cpp(g, v)$ to compute the maximum possible influence (e.g., posts, tweets) from one of the users in the seed community g to user v through some paths. Then, we provide the definition of the *influenced community* g^{Inf} below.

Definition 3 *(**Influenced Community,** g^{Inf}) Given a social network G, a seed community g, and an influence threshold θ ($\in [0, 1)$), the influenced community, g^{Inf}, of g is a subgraph of G, where each vertex v in $V(g^{Inf})$ satisfies the condition that $cpp(g, v) \geq \theta$.*

The Influential Score of the Seed Community. To evaluate the propagation effect from a seed community g to its influenced community g^{Inf}, we give the definition of the *influential score*, $\sigma(g)$, for the influenced community g^{Inf}:

$$\sigma(g) = \sum_{v \in V(g^{Inf})} cpp(g, v). \tag{5}$$

The influential score $\sigma(g)$ (given in Eq. (5)) sums up all the community-to-user propagation probabilities $cpp(g, v)$ from g to vertices v in the influenced community g^{Inf}. Intuitively, high influential score $\sigma(g)$ indicates that the seed community g may influence either a few users v with high community-to-user propagation probabilities $cpp(g, v)$, or many users v even with low $cpp(g, v)$ values.

The Problem of Top-L Most Influential Community Detection Over Social Networks. We are now ready to define the problem of detecting top-L communities with the highest influences.

Definition 4 *(**Top-L Most Influential Community Detection Over Social Networks, Top$L$-ICDE**) Given a social network G, a positive integer L, a threshold θ, a support, k, of the trusses, the maximum radius, r, of seed communities, and a set, Q, of query keywords, the problem of top-L most influential community detection over social networks (TopL-ICDE) retrieves L seed communities g_i, such that:*

- *g_i satisfy the constraints of seed communities (as given in Definition 2), and;*

- *these L seed communities g_i have the influenced communities, g_i^{Inf}, with the highest influential scores $\sigma(g_i)$,*

where $\sigma(g_i)$ is given by Eq. (5).

A Variant of Top$L$-ICDE (Diversified Top-L Most Influential Community Detection Over Social Networks). In Definition 4, the Top$L$-ICDE problem returns L individual seed communities with the highest influences. Note that, these L individual communities may influence the same users (i.e., with high overlaps of the influenced users). In order to achieve higher user impacts, in this paper, we also consider a variant of Top$L$-ICDE, namely *diversified top-L most influential community detection over social networks* (DTop$L$-ICDE), which obtains a set of L diversified communities with the highest collaborative influences on other users.

Given a set, S, of communities, we formally define its diversity score, $D(S)$, to evaluate the collective influence of communities in S on other users:

$$D(S) = \sum_{\forall v \in V(G)} \max_{\forall g \in S} \{cpp(g, v)\}. \quad (6)$$

The diversity score in Eq. (6) sums up the maximum possible community-to-user propagation probabilities, $cpp(g, v)$, from any community g in S to the influenced users v. Intuitively, a higher diversity score indicates higher impacts from communities in S.

For simplicity, we use $\Delta D_{g_i}(S)$ to represent the increment of the diversity score for adding the subgraph g_i to set S, i.e., $\Delta D_{g_i}(S) = D(S \cup \{g_i\}) - D(S)$.

Next, we define our DTop$L$-ICDE problem which returns L diversified seed communities with the highest diversity score (i.e., collaborative influences on other users).

Definition 5 ***(Diversified Top-L Most Influential Community Detection Over Social Networks, DTop$L$-ICDE)*** *Given a social network G, a positive integer L, a threshold θ, a support, k, of the trusses, the maximum radius, r, of seed communities, and a set, Q, of query keywords, the problem of diversified top-L most influential community detection over social networks (DTop$L$-ICDE) retrieves a set, S of L seed communities g_i, such that:*

- *g_i satisfy the constraints of seed communities (as given in Definition 2), and;*
- *the set S of L seed communities g_i has the highest diversity score $D(S)$ (as given by Eq. (6)).*

Intuitively, the DTop$L$-ICDE problem (given in Definition 5) finds a set, S, of L communities that have the highest collaborative influence, that is, the diversity score $D(S)$ in Eq. (6), which is defined as the summed influence from communities g in S to the influenced users.

Challenges. A straightforward method to tackle the Top$L$-ICDE problem is to first obtain all possible seed communities (subgraphs) of the data graph G, then check the constraints of these seed communities, and finally rank these seed communities based on their influential scores. Similarly, for DTop$L$-ICDE, we can also compute the diversity score for any combination of L communities, and choose a set of size L with the highest diversity score. However, such straightforward methods are quite inefficient, especially for large-scale social networks, due to the high costs of the constraint checking over an exponential number of possible seed communities (or community combinations), as well as the costly computation of influence/diversity scores (as given in Eqs. (5) and (6)). Thus, the processing of the Top$L$-ICDE problem (and its variant DTop$L$-ICDE) raises up the efficiency and scalability issues for detecting (diversified) top-L most influential communities over large-scale social networks.

III. Our Top$L$-ICDE Processing Framework

Algorithm 1 presents our framework to efficiently answer Top$L$-ICDE queries, which consists of two phases, i.e., *offline pre-computation* and *online TopL-ICDE processing phases*.

In the offline pre-computation phase, we pre-process the social network graph based on pre-computing data (e.g., influential score bounds) to facilitate online query answering and constructing an index over these pre-computed data. In particular, for each vertex v_i in data graph G, we first hash its associated keyword set $v_i.W$ into a bit vector $v_i.BV$ (lines 1-2). Then, for each r-radius subgraph $hop(v_i, r)$ centered at vertex v_i and with a radius $r \in [1, r_{max}]$, we offline pre-compute data (e.g., support/influence bounds) to facilitate the pruning (lines 3-5). Next, we construct a tree index $\mathcal{I}$ over the pre-computed data (line 6).

In the online Top$L$-ICDE processing phase, for each query, we traverse the index $\mathcal{I}$ to efficiently retrieve candidate seed communities, by integrating our proposed effective pruning strategies (i.e., keyword, support, radius, and influential score) (lines 7-8). Finally, we refine these candidate seed communities by computing their actual influential scores, and return the top-L most influential communities with the highest influential scores (line 9).

Discussions on the DTop$L$-ICDE Framework. We will discuss the DTop$L$-ICDE framework later in Section VII, which follows the Top$L$-ICDE framework but applies specifically designed pruning/refinement techniques to retrieve L diversified communities (i.e., DTop$L$-ICDE query answers).

IV. Pruning Strategies

In this section, we present effective pruning strategies to reduce the Top$L$-ICDE problem search space in our framework (line 8 of Algorithm 1). **Due to space limitations, in subsequent discussions, we will omit proofs of all the lemmas, which can be found in our technical report [17].**

A. Keyword Pruning

In this subsection, we first provide an effective *keyword pruning* method. From Definition 2, any vertex in the seed community g must contain at least one query keyword in Q. Thus, our keyword pruning method filters out those candidate subgraphs g containing some vertices without query keywords.

Lemma 1 (Keyword Pruning) *Given a set, Q, of query keywords and a candidate subgraph g, subgraph g can be*

Algorithm 1: The Top$L$-ICDE Process Framework

Input: i) a social network G; ii) a set, Q, of query keywords; iii) the support, k, of the truss for each seed community; iv) the maximum radius, r, of seed communities; v) the influence threshold θ, and; vi) integer parameter L
Output: a set, S, of top-L seed communities
// **offline pre-computation phase**
for *each* $v_i \in V(G)$ **do**
  hash keywords in $v_i.W$ into a bit vector $v_i.BV$;
  for $r = 1$ **to** r_{max} **do**
    extract r-hop subgraph $hop(v_i, r)$ of vertex v_i;
    offline pre-compute data, $v_i.R$, w.r.t. the support upper bound $ub\_sup(\cdot)$ and influence upper bound Inf_{ub} for subgraph $hop(v_i, r)$;
build a tree index $\mathcal{I}$ over graph G with pre-computed data as aggregates;
// **online Top$L$-ICDE processing phase**
for *each TopL-ICDE query* **do**
  traverse the tree index $\mathcal{I}$ by applying *keyword*, *support*, *radius*, and *influential score pruning* strategies to retrieve candidate seed communities;
  refine candidate seed communities to obtain top-L seed communities with the highest influential scores;

safely pruned, if there exists at least one vertex $v_i \in V(g)$ such that: $v_i.W \cap Q = \emptyset$ holds, where $v_i.W$ is the keyword set associated with vertex v_i.

B. Support Pruning

According to Definition 2, the seed community g should be a k-truss [16], that is, the support $sup(e_{u,v})$ of each edge $e_{u,v} \in E(g)$ (defined as the number of triangles that contain edge $e_{u,v}$ in the seed community g) must be at least $(k-2)$.

Assume that we can offline obtain an upper bound, $ub\_sup(e_{u,v})$, of the support $sup(e_{u,v})$ on edge $e_{u,v}$ in g. Then, we have the following lemma to discard those candidate seed communities g containing some edges with low support.

Lemma 2 (Support Pruning) *Given a seed community g and a parameter k, subgraph g can be safely pruned if there exists an edge $e_{u,v} \in E(g)$ satisfying $ub\_sup(e_{u,v}) < (k-2)$.*

Discussions on How to Obtain the Support Upper Bound $\mathbf{ub\_sup(e_{u,v})}$. To enable the support pruning, we need to calculate the support upper bound $ub\_sup(e_{u,v})$ of edge $e_{u,v}$ in a seed community g. Since the seed community g is a subgraph of the data graph G, the support of edge $e_{u,v}$ in g is thus smaller than or equal to that in G (in other words, the number of triangles containing $e_{u,v}$ in g is less than or equal to that in G). Therefore, we can use the edge support in the data graph G (or any supergraph of g) as the support upper bound $ub\_sup(e_{u,v})$ of edge $e_{u,v}$.

C. Radius Pruning

From Definition 4, the maximum radii, r, of seed communities are online specified by users, which limits the shortest path distance between the center vertex and any other vertices to be not more than r. We provide the following pruning lemma with respect to radius r. If a subgraph g (centered at vertex v_i) has a radius greater than r, it violates the radius constraint of the seed community.

Lemma 3 (Radius Pruning) *Given a subgraph g (centered at vertex v_i) and the maximum radius, r, of seed communities, subgraph g can be safely pruned, if there exists a vertex $v_l \in V(g)$ such that $dist(v_i, v_l) > r$, where function $dist(x, y)$ outputs the number of hops between vertices x and y in g.*

This radius pruning method enables offline pre-computation by extracting subgraphs for any possible radius $r \in [1, r_{max}]$, that is, $hop(v_i, r)$. In other words, those vertices with distance to vertex v_i greater than radius r can be safely ignored, so we only need to focus on r-hop subgraphs $hop(v_i, r)$.

D. Influential Score Pruning

Next, we discuss the *influential score pruning* method below, which filters out those seed communities with low influential scores.

Lemma 4 (Influential Score Pruning) *Assume that we have obtained L seed communities g_i so far, and let σ_L be the smallest influential score among these L seed communities g_i. Any subgraph g can be safely pruned, if it holds that $ub\_\sigma(g) \leq \sigma_L$, where $ub\_\sigma(g)$ is the upper bound of the influential score $\sigma(g)$.*

Discussions on How to Obtain the Upper Bound $ub\_\sigma(g)$ of the Influential Score. Based on Eq. (5), the influential score $\sigma(g)$ of seed community g is given by summing up the community-to-user propagation probabilities $cpp(g, v)$, for $v \in g^{Inf}$, where $cpp(g, v) \geq \theta$. Since threshold θ is online specified by the user, we can offline pre-select m thresholds θ_1, θ_2, ..., and θ_m (assuming $\theta_1 < \theta_2 < ... < \theta_m$), and pre-calculate the influential scores $\sigma_z(g)$ w.r.t. thresholds θ_z (for $1 \leq z \leq m$). Given an online threshold θ, if $\theta \in [\theta_z, \theta_{z+1})$ holds, we will use $\sigma_z(g)$ as the influential score upper bound $ub\_\sigma(g)$, where $\sigma_z(g)$ is the influential score of g using threshold θ_z.

V. Offline Pre-Computation and Indexing

In this section, we discuss how to offline pre-compute data for social networks to facilitate effective pruning in Section V-A, and construct indexes over these pre-computed data to help with online Top$L$-ICDE processing in Section V-B.

A. Offline Pre-Computed Data for TopL-ICDE Processing

In order to facilitate online Top$L$-ICDE processing, we perform offline pre-computations over data graph G and obtain some aggregate information of potential seed communities, which can be used for our proposed pruning strategies to reduce the problem search space. Specifically, for each vertex $v_i \in V(G)$, we first hash the keyword set $v_i.W$ into a bit vector $v_i.BV$ of size B. Each edge $e_{u,v}$ is associated with the edge support upper bound $ub\_sup(e_{u,v})$ in r_{max}-hop subgraph $hop(v_i, r_{max})$ (i.e., a pair of $(v_i, ub\_sup(e_{u,v}))$). Next, we use the radius pruning (as given in Lemma 3) to enable offline pre-computations of r-hop subgraphs. In particular, starting from each vertex v_i, we traverse the data graph G in a breadth-first manner (i.e., BFS), and obtain r-hop subgraphs, $hop(v_i, r)$, centered at v_i with radii $r \in [1, r_{max}]$. For each

r-hop subgraph, we calculate and store aggregates in a list $v_i.R$, in the form $(v_i.BV_r, v_i.ub\_sup_r, [(\sigma_z, \theta_z)])$ as follows:

- a bit vector, $v_i.BV_r$, which is obtained by hashing all keywords in the keyword sets $v_l.W$ of vertices v_l in the r-hop subgraph $hop(v_i, r)$ into a position in the bit vector (i.e., $v_i.BV_r = \bigvee_{\forall v_l \in V(hop(v_i,r))} v_l.BV$);
- an upper bound, $v_i.ub\_sup_r$, of all support bounds $ub\_sup(e_{u,v})$ for edges $e_{u,v}$ in the r-hop subgraph $hop(v_i, r)$ (i.e., $v_i.ub\_sup_r = \max_{\forall e_{u,v} \in E(hop(v_i,r))} ub\_sup(e_{u,v})$), and;
- m pairs of influential score upper bounds and influence thresholds $(\sigma_z(hop(v_i, r)), \theta_z)$ (for $1 \leq z \leq m$).

More details of the aggregates are provided below:

The Computation of Keyword Bit Vectors $v_i.BV_r$: We first obtain the keyword bit vector $v_i.BV$ of size B for each vertex $v_i \in V(G)$, and then compute the one, $v_i.BV_r$, for r-hop subgraph $hop(v_i, r)$. Specifically, for each vertex v_i, we first initialize all bits in a vector $v_i.BV$ with zeros. Then, for each keyword w in the keyword set $v_i.W$, we use a hashing function $f(w)$ that maps a keyword w to an integer between $[0, B-1]$ and set the $f(w)$-th bit position to 1 (i.e., $v_i.BV[f(w)] = 1$). Next, for all vertices v_l in r-hop subgraph $hop(v_i, r)$, we perform a bit-OR operator over their bit vectors $v_l.BV$. That is, we have $v_i.BV_r = \bigvee_{\forall v_l \in V(hop(v_i,r))} v_l.BV$.

The Computation of Support Upper Bounds $v_i.ub\_sup_r$: For each radius $r \in [1, r_{max}]$, we compute a support upper bound $v_i.ub\_sup_r$ as follows. We first calculate the support upper bound, $ub\_sup(e_{u,v})$, for each edge $e_{u,v}$ in the r_{max}-hop subgraph $hop(v_i, r_{max})$. Then, the maximum $ub\_sup(e_{u,v})$ among all edges in $hop(v_i, r_{max})$ is selected as the support upper bound $v_i.ub\_sup_r$.

The Computation of Influential Score Upper Bounds $\sigma_z(hop(v_i, r))$ (w.r.t. θ_z): Since seed communities g are subgraphs of some r-hop subgraphs $hop(v_i, r)$, as given in Eq. (5), the influential score $\sigma_z(hop(v_i, r))$ (w.r.t. influence threshold θ_z) must be greater than or equal to $\sigma_z(g)$ and is thus an influential score upper bound. In other words, we overestimate the influence of a seed community g inside r-hop subgraph $hop(v_i, r)$, by assuming that $g = hop(v_i, r)$.

To calculate the influential score $\sigma_z(hop(v_i, r))$, we first start from each r-hop subgraph $hop(v_i, r)$ $(= g)$ and then expand $hop(v_i, r)$ to obtain its influenced community g^{Inf} via a graph traversal. A vertex u is included in g^{Inf}, if it holds that $cpp(g, u) \geq \theta_z$, where $cpp(g, u)$ is given by Eq. (4). The graph traversal algorithm terminates, when $cpp(g, u) < \theta_z$ holds. Finally, we use Eq. (5) to calculate the influential score $\sigma_z(hop(v_i, r))$ of the expanded graph (w.r.t., θ_z).

Offline Computation Algorithm: Algorithm 2 illustrates the epseudo code of offline data pre-computation in the data graph G that can facilitate online Top$L$-ICDE processing. In particular, for each vertex $v_i \in V(G)$, all the keywords of the keyword set $v_i.W$ are hashed into a bit vector $v_i.BV$ and stored in a list $v_i.R$ (lines 1-3). Then, we compute the support for each edge $e_{u,v}$ in $hop(v_i, r_{max})$ and use the maximum one as the support upper bound $ub\_sup(e_{u,v})$ for the edge $e_{u,v}$ (lines 4-5). Next, for each vertex v_i and pre-selected radius r ranging from 1 to r_{max}, we calculate the pre-computed data for subgraph $hop(v_i, r)$, including keyword bit vector $v_i.BV_r$ (lines 6-8), edge support upper bound $v_i.ub\_sup_r$ (line 9), and upper bound of influential score $\sigma_z(hop(v_i, r))$ w.r.t. θ_z (lines 10-11). All these pre-computed data are added to the list $v_i.R$ (lines 12-13), which is returned as the output (line 14).

Algorithm 2: Offline Pre-Computation

Input: i) a social network G; ii) the maximum value of radius r_{max}; iii) m influence thresholds $\{\theta_1, \theta_2, \ldots, \theta_m\}$;
Output: pre-computed data $v_i.R$ for each vertex v_i;

```
1  for each v_i ∈ V(G) do
       // keyword bit vectors
2      hash all keywords in the keyword set v_i.W into a bit vector v_i.BV;
3      v_i.R = {v_i.BV};
       // edge support upper bounds
4      for each e_{u,v} ∈ E(hop(v_i, r_max)) do
5          compute edge support upper bounds ub_sup(e_{u,v}) w.r.t. hop(v_i, r_max);
6  for each v_i ∈ V(G) do
7      for each r = 1 to r_max do
8          v_i.BV_r = ⋁_{∀v_l ∈ V(hop(v_i,r))} v_l.BV;
9          v_i.ub_sup_r = max_{∀e_{u,v} ∈ E(hop(v_i,r))} ub_sup(e_{u,v});
           // influential score upper bounds
10         for each θ_z ∈ {θ_1, θ_2, ..., θ_m} do
11             σ_z(hop(v_i, r)) = calculate_influence(hop(v_i, r), θ_z);
12             add (σ_z(hop(v_i, r)), θ_z) to v_i.R;
13         add v_i.BV_r and v_i.ub_sup_r to v_i.R;
14 return v_i.R;
```

Complexity Analysis: For Algorithm 2, the time complexity is given by $O(|V(G)| \cdot (|W| + avg\_deg^{r_{max}} + r_{max} \cdot ((B + 1)avg\_deg^{r-1} + m \cdot ((|E(g_r^{inf})| + |V(g_r^{inf})|)log|V(g_r^{inf})|))))$, where avg_deg denote the average number of vertex degree and g_r^{inf} is the subgraph containing the influenced users from $hop(v_i, r)$. The space complexity is $O(|V(G)| + |E(G)| + |V(G)| \cdot r_{max} \cdot (B + 2m + 1))$. Please see the detailed descriptions for Algorithm 2 in our technical report [17].

B. Indexing Mechanism

In this subsection, we illustrate the details of building a tree index, $\mathcal{I}$, over pre-computed data of social networks G, which can be used for performing online Top$L$-ICDE processing.

The Data Structure of Index $\mathcal{I}$: We will construct a hierarchical tree index, $\mathcal{I}$, over social networks G, where each index node N contains multiple entries N_i, each corresponding to a subgraph of G.

Specifically, the tree index $\mathcal{I}$ contains two types of nodes, leaf and non-leaf nodes.

Leaf Nodes: Each leaf node N in index $\mathcal{I}$ contains multiple vertices $v_i \in V(G)$. Each vertex v_i is associated with the following pre-computed data in $v_i.R$ (w.r.t. each possible radius $r \in [1, r_{max}]$).

- a keyword bit vector $v_i.BV_r$;
- edge support upper bound $v_i.ub\_sup_r$, and;
- m pairs of influential score upper bounds and influence thresholds $(\sigma_z(hop(v_i, r)), \theta_z)$.

Non-Leaf Nodes: Each non-leaf node N in index $\mathcal{I}$ has multiple index entries, each of which, N_i, is associated with the following aggregate data below (w.r.t. each radius $r \in [1, r_{max}]$).

- an aggregated keyword bit vector $N_i.BV_r = \bigvee_{\forall v_l \in N_i} v_l.BV_r$;
- the maximum edge support upper bound $N_i.ub\_sup_r = \max_{\forall v_l \in N_i} v_l.ub\_sup_r$;
- m pairs, $(N_i.\sigma_z, \theta_z)$, of maximum influential score upper bounds and influence thresholds (for $N_i.\sigma_z = \max_{\forall v_l \in N_i} \sigma_z(hop(v_l, r))$), and;
- a pointer, $N_i.ptr$, pointing to a child node.

Index Construction: To construct the tree index $\mathcal{I}$, we sorted all vertices by their average of ub_sup_r and σ_z and recursively divided sorted vertices array into partitions of the similar sizes, and then obtain index nodes on different levels of the tree index. Then, we associate each index entry in non-leaf nodes (or each vertex in leaf nodes) with its corresponding aggregates (or pre-computed data).

Complexity Analysis: The time complexity of our tree index construction is given by $O((\gamma^{\lceil \log_\gamma |V(G)| \rceil + 1} - 1)/(\gamma - 1))$, where γ is the average fanout of each non-leaf node N. The space complexity is given by $O(r_{max} \cdot (B + 2m + \gamma) \cdot ((\gamma^{\lceil \log_\gamma |V(G)| \rceil} - 1)/(\gamma - 1)) + |V(G)| \cdot (B + 2m))$. Due to space limitations, please refer to our technical report [17] for the detailed description.

VI. Online Top$L$-ICDE Processing

For the online Top$L$-ICDE processing phase (see Algorithm 1), we utilize the constructed tree indexes to conduct the Top$L$-ICDE processing, by integrating our effective pruning strategies and returning top-L most influential seed communities.

A. Index Pruning

In this subsection, we provide effective pruning heuristics on index nodes, which can filter out all candidate seed communities under index nodes. Proofs of lemmas are omitted here due to space limitations.

Keyword Pruning for Index Entries: We utilize the aggregated keyword bit vector, $N_i.BV_r$, of an index entry N_i, and discard an index entry N_i if none of r-hop subgraphs under N_i contain some keyword in the query keyword set Q.

Lemma 5 (Index-Level Keyword Pruning) *Given an index entry N_i and a set, Q, of query keywords, entry N_i can be safely pruned, if it holds that $N_i.BV_r \wedge Q.BV = \mathbf{0}$, where $Q.BV$ is a bit vector hashed from the query keyword set Q.*

Support Pruning for Index Entries: We next use the support parameter k in the k-truss constraint (as given in Definition 2) to prune an index entry N_i with low edge supports.

Lemma 6 (Index-Level Support Pruning) *Given an index entry N_i and a support parameter k, entry N_i can be safely pruned, if it holds that $N_i.ub\_sup_r < k$, where $N_i.ub\_sup_r$ is the maximum edge support upper bound in all r-hop subgraphs under N_i.*

Influential Score Pruning for Index Entries: Since the Top$L$-ICDE problem finds L seed communities with the highest influential scores, we can employ the following pruning lemma to rule out an index entry N_i whose influential score upper bound lower than that of L candidate seed communities we have seen so far.

Lemma 7 (Index-Level Influential Score Pruning) *Assume that we have obtained L candidate seed communities with the smallest influential score σ_L. Given an index entry N_i and an influence threshold $\theta \in [\theta_z, \theta_{z+1})$, entry N_i can be safely pruned, if it holds that $N_i.\sigma_z \leq \sigma_L$.*

B. TopL-ICDE Processing Algorithm

Algorithm 3: Online Top$L$-ICDE Processing

Input: i) a social network G; ii) a set, Q, of query keywords; iii) the support, k, of the truss for each seed community; iv) the maximum radius, r, of seed communities; v) the influence threshold $\theta \in [\theta_z, \theta_{z+1})$; vi) an integer parameter L, and; vii) a tree index $\mathcal{I}$ over G

Output: a set, S, of top-L most influential communities

```
// initialization
1  hash all keywords in the query keyword set Q into a query bit vector Q.BV;
2  initialize a maximum heap H in the form of (N, key);
3  insert (root(I), 0) into heap H;
4  S = ∅; cnt = 0; σ_L = −∞;
   // index traversal
5  while H is not empty do
6      (N, key) = de-heap(H);
7      if key ≤ σ_L then
8          terminate the loop;
9      if N is a leaf node then
10         for each vertex v_i ∈ N do
11             if r-hop subgraph hop(v_i, r) cannot be pruned by
                 Lemma 1, 2, or 4 then
12                 obtain seed communities g ⊆ hop(v_i, r)
                     satisfying the constraints;
13                 compute influential score σ(g) =
                     calculate_influence(g, θ);
14                 if cnt < L then
15                     add (g, σ(g)) to S;
16                     cnt = cnt + 1;
17                     if cnt = L then
18                         set σ_L to the smallest influential score
                             in S;
19                 else
20                     if σ(g) > σ_L then
21                         add (g, σ(g)) to S;
22                         remove a candidate seed community
                             with the lowest influential score from
                             S;
23                         update influence threshold σ_L;
24     else // N is a non-leaf node
25         for each entry N_i ∈ N do
26             if N_i cannot be pruned by Lemma 5, 6, or 7 then
27                 insert entry (N_i, N_i.σ_z) into heap H;
28 return S;
```

Algorithm 3 gives the pseudo-code to answer a Top$L$-ICDE query over a social network G via the index $\mathcal{I}$. Specifically, the algorithm first initializes some data structure/variables (lines 1-4), then traverses the index $\mathcal{I}$ (lines 5-27), and finally returns actual Top$L$-ICDE query answers (line 28).

Initialization: Given a query keyword set Q, we first hash all the keywords in Q into a query bit vector $Q.BV$ (line 1). Then, we use a *maximum heap* $\mathcal{H}$ to traverse the index, which contains heap entries in the form of (N, key), where

N is an index node and key is the key of node N (defined as maximum influential score upper bound $N.\sigma_z$, mentioned in Section V-B). Intuitively, if a node N has a higher influential score upper bound, it is more likely that N contains seed communities with high influential scores (ranks). We thus always use the maximum heap $\mathcal{H}$ to access nodes with higher influential scores earlier. We initialize heap $\mathcal{H}$ by inserting the index root in the form $(root(\mathcal{I}), 0)$ (lines 2-3). In addition, we use a result set, S, to store candidate seed communities (initialized with an empty set) whose entries are in the form of $(g, \sigma(g))$, a variable cnt (initially set to 0) to record the size of set S, and an influence threshold σ_L (w.r.t. S, initialized to $-\infty$) (line 4).

Index Traversal: Next, we employ the maximum heap $\mathcal{H}$ to traverse the index $\mathcal{I}$ (lines 5-27). Each time we pop out an entry (N, key) with node N and the maximum key, key, in the heap (lines 5-6). If key is not greater than the smallest influential score, σ_L, among L seed communities in S, then all entries in heap $\mathcal{H}$ have influential score upper bounds not greater than σ_L. Therefore, we can safely prune the remaining (unvisited) entries in the heap and terminate the index traversal (lines 7-8). When we encounter a leaf node N, we consider r-hop subgraphs $hop(v_i, r)$ for all vertices v_i under node N (lines 9-10). Then, we apply the community-level pruning strategies, *keyword pruning*, *support pruning*, and *influential score pruning*. If an r-hop subgraph $hop(v_i, r)$ cannot be pruned, we obtain seed communities, g, within $hop(v_i, r)$ that satisfy the constraints given in Definition 2 and compute their accurate influential scores $\sigma(g)$ w.r.t. threshold θ, by invoking the function calculate_influence(g, θ) (lines 11-13). Then, we will update the result set S with g, by considering the following two cases.

Case 1: If the size, cnt, of set S is less than L, a new entry $(g, \sigma(g))$ will be added to S (lines 14-16). If the set size cnt reaches L, we will set σ_L to the smallest influential score in S (lines 17-18).

Case 2: If the size, cnt, of set S is equal to L and the influential score $\sigma(g)$ is greater than σ_L, we will add the new entry $(g, \sigma(g))$ to S, and remove a seed community with the lowest influential score from S (lines 19-22). Accordingly, threshold σ_L will be updated with the new set S (line 23).

On the other hand, when we visit a non-leaf node N, for each child entry $N_i \in N$, we will apply the index-level pruning strategies, including *index-level keyword pruning*, *index-level support pruning*, and *index-level influential score pruning* (lines 24-26). If N_i cannot be pruned, we insert a heap entry $(N_i, N_i.\sigma_z)$ into heap $\mathcal{H}$ for further investigation (line 27).

Finally, after the index traversal, we return actual Top$L$-ICDE query answers in S (line 28).

Discussions on the Influential Score Calculation Function calculate_influence(g, θ)**:** To calculate the influential score (via Eqs. (4) and (5)), we need to obtain the influenced community g^{Inf} from seed community g, whose process is similar to the single-source shortest path algorithm. We will first compute 1-hop neighbors, v_k, of boundary vertices in seed community g, and include v_k (satisfying $cpp(g, v_k) \geq \theta$) in the influenced community g^{Inf}. Then, each time we expand one hop from the current influenced community g^{Inf}, by adding to g^{Inf} new vertices v_{new} if it holds that $cpp(g, v_{new}) \geq \theta$, where $cpp(g, v_{new}) = \max_{\forall u \in V(g^{Inf})}\{upp(u, v_{new})\} = \max_{\forall u \in V(g^{Inf})}\{cpp(g, u) \cdot p_{u,v_{new}}\}$.

Top$L$-ICDE Complexity Analysis: The time complexity of Algorithm 3 is given by $O(\sum_{j=1}^{h} f^{h-j+1} \cdot (1 - PP^{(j)}) + f^{h+1} \cdot (1 - PP^{(0)}) \cdot \overline{n_r} \cdot (|E(g^{Inf}| + |V(g^{Inf})|) log |V(g^{Inf})|))$. Due to space limitations, please refer to our technical report [17] for detailed descriptions.

VII. Online Diversified Top$L$-ICDE Processing

In this section, we discuss how to efficiently tackle the Top$L$-ICDE variant, that is, the DTop$L$-ICDE problem.

A. NP-Hardness of the DTopL-ICDE Problem

First, we prove the NP-hardness of our DTop$L$-ICDE problem (given in Definition 5) in the following lemma.

Lemma 8 *The DTopL-ICDE problem is NP-hard.*

To prove Lemma 8 (i.e., DTop$L$-ICDE is NP-hard), we can reduce a known NP-hard problem, the *Maximum Coverage* problem [18], to our DTop$L$-ICDE problem. Please refer to the details of the proof in our technical report [17].

B. The Greedy Algorithm for DTopL-ICDE Processing

Due to its NP-hardness (as given by Lemma 8), our DTop$L$-ICDE problem is not tractable. Therefore, alternatively, we will propose a greedy algorithm to process the DTop$L$-ICDE query with an approximation bound.

A Framework for the DTop$L$-ICDE Greedy Algorithm. Our greedy algorithm has two steps. First, we invoke *online TopL-ICDE processing algorithm* (Algorithm 3) to obtain a set, T, of top-(nL) candidate communities with the highest influence scores, where n (> 1) is a user-specified parameter. Intuitively, communities with high influences are more likely to contribute to the DTop$L$-ICDE community set S with high diversity scores.

Next, we will identify L out of these (nL) candidate communities in T with high diversity score (forming a set S of size L). To achieve this, we give a naive method of our greedy algorithm without any pruning, denoted as Greedy_WoP, as follows. Given (nL) candidate communities with the highest influence scores in a set T, we first add the candidate community g in T with the highest influence to S (removing g from T). Then, each time we select one candidate community $g \in T$ with the highest diversity score increment $\Delta_g(S)$, among all communities in T, and move g from T to S. This process repeats until L communities are added to S.

Effective Pruning Strategy w.r.t. Diversity Score. In the greedy algorithm without any pruning Greedy_WoP, we have to check all the nL candidate communities in T, compute their diversity score increments $\Delta_{g_m}(S)$, and select the one with the highest diversity score increment, which is quite costly with the time complexity $O(nL^2)$.

To reduce the search space, we will propose an effective *diversity score pruning* method, which can avoid scanning all

communities in T in each round (i.e., those communities with low diversity score increments can be safely pruned).

Before introducing our pruning strategy, we will first give two properties of the diversity score $D(S)$ below:

- **Monotonicity:** given two subgraph sets S and S', satisfying that $S' \subseteq S$, it holds that $D(S') \leq D(S)$, and;
- **Submodularity:** given two subgraph sets S and S' and a subgraph g, satisfying that $S' \subseteq S$ and $g \notin S'$, it holds that $D(S' \cup \{g\}) - D(S') \geq D(S \cup \{g\}) - D(S)$ (i.e., $\Delta D_g(S') \geq \Delta D_g(S)$).

By utilizing the two properties above, we have the following pruning lemma:

Lemma 9 (Diversity Score Pruning) *Assume that we have a set, T, of candidate seed communities g, and a set, S, of the currently selected DTopL-ICDE answers. Given a subset $S' \subseteq S$ and a subgraph $g \in T$ with the diversity score increment $\Delta D_g(S)$, any subgraph $g_m \in T$ can be safely pruned, if it holds that $ub\_\Delta D_{g_m}(S) < \Delta D_g(S)$, where $ub\_\Delta D_{g_m}(S)$ is an upper bound of the diversity score increment $\Delta D_{g_m}(S)$ (which can equal to either $\Delta D_{g_m}(S')$ or $\sigma(g_m)$).*

The DTop$L$-ICDE Greedy Algorithm with Pruning, Greedy_WP. Algorithm 4 shows the pseudo-code to handle the online DTop$L$-ICDE query over a given social network G. Specifically, the algorithm invokes *online TopL-ICDE processing algorithm* (i.e., Algorithm 3) to obtain a set, T, of nL candidate communities with the highest influences (line 1), then refines the set T to obtain L communities with the highest diversity score (lines 2-15), and finally returns the actual DTop$L$-ICDE answers (line 16).

Initialization: Specifically, after obtaining nL candidate communities via Algorithm 3 (line 1), we initialize a *maximum heap*, $\mathcal{H}$, that stores entries in the form (g, key_g), where g is a seed community and key_g is the key of the heap entry (defined as the upper bound, $ub\_\Delta D_g(S)$, of the diversity score increment) (line 2). For each candidate community $g \in T$, we set its round number, $g.round$, to 0, and the upper bound $\sigma(g)$ of the diversity score increment in this round. Then, we insert entries $(g, \sigma(g))$ into heap $\mathcal{H}$ for refinement (lines 3-5). We also maintain an initially empty answer set S, and set initial round number, $round$, to 0 (line 6).

Candidate Community Refinement: To refine candidate communities in heap $\mathcal{H}$, each time we pop out an entry $(g, ub\_\Delta D_g(S))$ from $\mathcal{H}$ with the maximum key (line 8), and check whether or not the key $ub\_\Delta D_g(S)$ is computed at this round (i.e., $g.round = round$), considering the following two cases (lines 9-15):

Case 1: If $g.round = round$ holds, it indicates that g is the one with the highest diversity score increment in the current round, $round$ (as proved by Lemma 9). Thus, g will be added to the DTop$L$-ICDE answer set S and $round$ is increased by 1 (lines 9-11).

Case 2: If $g.round \neq round$ holds (line 12), the entry key, $ub\_\Delta D_g(S)$, is outdated, which is equal to $\Delta D_g(S')$ ($S' \subseteq S$). Thus, we will re-compute the diversity score increment, $\Delta D_g(S)$, w.r.t. the current answer set S, update $g.round$ with $round$, and insert $(g, \Delta D_g(S))$ back into $\mathcal{H}$ (lines 13-15).

After picking L candidates from $\mathcal{H}$ to S, the algorithm terminates the loop (line 7) and returns S as DTop$L$-ICDE answers (line 16).

Algorithm 4: Online DTop$L$-ICDE Processing

Input: i) a social network G; ii) a set, Q, of query keywords; iii) the support, k, of the truss for each seed community; iv) the maximum radius, r, of seed communities; v) the influence threshold $\theta \in [\theta_z, \theta_{z+1})$; vi) two integer parameters n and L, and; vii) a tree index $\mathcal{I}$ over G

Output: a set, S, of diversified top-L most influential communities

```
// obtain (nL) DTopL-ICDE candidates
```
1 invoke online Top$L$-ICDE processing algorithm (Algorithm 3) to obtain a set, T, of top-(nL) most influential seed communities;

```
// refine candidates via Greedy_WP
```
2 initialize a maximum heap $\mathcal{H}$ with entries in the form of (g, key_g);
3 **for** *each candidate seed community* $g \in T$ **do**
4 set $g.round = 0$;
5 insert $(g, \sigma(g))$ into heap $\mathcal{H}$;
6 $S = \emptyset$, $round = 0$;
7 **while** $|S| < L$ **do**
8 $(g, ub\_\Delta D_g(S))$ = de-heap($\mathcal{H}$);
9 **if** $g.round = round$ **then**
10 add g to S;
11 $round = round + 1$;
12 **else**
13 compute the increment of the diversity score $\Delta D_g(S) = D(S \cup \{g\}) - D(S)$;
14 $g.round = round$;
15 insert $(g, \Delta D_g(S))$ into heap $\mathcal{H}$;
16 **return** S;

DTop$L$-ICDE Complexity Analysis: The time complexity of Algorithm 4 is given by $O\left(\sum_{k=1}^{L}(nL - k + 1) \cdot (1 - DPP^{(k)}) \cdot |\bigcup_{g \in T} V(g^{Inf})|\right)$. Due to space limitations, please refer to our technical report [17] for detailed descriptions.

Approximation Ratio Analysis: From [14], for a function following *monotonicity* and *submodularity* properties, the approximate greedy algorithm has a $(1 - 1/e)$ approximation guarantee. Moreover, we can prove that our greedy algorithm has a $\epsilon \cdot (1 - 1/e)$ approximation guarantee, where $0 < \epsilon \leq 1$.

Lemma 10 *The online DTopL-ICDE processing algorithm can process the DTopL-ICDE query approximately within better than a factor of $\epsilon \cdot (1 - 1/e)$, where $0 < \epsilon \leq 1$.*

VIII. Experimental Evaluation

A. Experimental Settings

We tested the performance of our Top$L$-ICDE processing approach (i.e., Algorithm 3) on both real and synthetic graphs.

Real-World Graphs: We used two real-world graphs, *DBLP* and *Amazon*, similar to previous works [19]–[21], whose statistics are depicted in Table II. *DBLP* is a bibliographical network, in which two authors are connected if they co-authored at least one paper, whereas *Amazon* is an *Also Bought* network where two products are connected if they are co-purchased by customers.

Synthetic Graphs: For synthetic social networks, we generate *Newman–Watts–Strogatz small-world* graphs G [22]. Specifically, we first produce a ring with size $|V(G)|$, and then

TABLE II
STATISTICS OF REAL-WORLD GRAPH DATA SETS *DBLP* AND *Amazon*.

Social Networks	$\|V(G)\|$	$\|E(G)\|$
DBLP	317,080	1,049,866
Amazon	334,863	925,872

TABLE III
PARAMETER SETTINGS.

Parameters	Values
support, k, of truss structure	3, **4**, 5
radius r	1, **2**, 3
size, L, of query result set	2, 3, **5**, 8, 10
the size, $\|V(G)\|$, of data graph G	10K, 25K, **50K**, 100K, 250K, 500K, 1M
parameter, n, for DTop$L$-ICDE	2, 3, **5**, 8, 10

connect each vertex with its m nearest neighbors in the ring. Next, for each resulting edge $e_{u,v}$, with probability μ, we add a new edge $e_{u,w}$ between u and a random vertex w. Here, we set $m = 6$ and $\mu = 0.167$. For each vertex, we also randomly produce a keyword set $v_i.W$ from the keyword domain Σ, following Uniform, Gaussian, or Zipf distribution, and obtain three synthetic graphs, denoted as *Uni*, *Gau*, and *Zipf*, respectively. For each edge $e_{u,v}$ in graph G, we randomly generate a value within the interval $[0.5, 0.6)$ as the edge weight $p_{u,v}$. The propagation probabilities can be computed based on the MIA model (Section II-B).

In our experiments, we randomly select $|Q|$ keywords from the keyword domain Σ and form a query keyword set Q.

Competitor: To our best knowledge, no prior works studied the Top$L$-ICDE problem and its variant DTop$L$-ICDE problem by considering highly connected k-truss communities with user-specified keywords and high (collective) influences on other users. Thus, for Top$L$-ICDE, we use a baseline method, named *ATindex*, which applies the state-of-the-art (k, d)-truss community search algorithm [23]. Specifically, *ATindex* offline pre-computes and indexes the trussness on vertices and edges. Then, it online filters out vertices with trussness less than k via the index, extracts r-hop subgraphs (satisfying the keyword constraints w.r.t. Q) centered at the remaining vertices, and obtains maximal k-truss within r-hop subgraphs. After that, *ATindex* computes influential scores of these k-truss subgraphs and returns L communities with the highest influential scores.

For DTop$L$-ICDE, we compare our approach (using Greedy_WP) with Greedy_WoP and *Optimal* methods. Greedy_WoP is the greedy algorithm without pruning mentioned in Section II-C, whereas *Optimal* computes the diversity score for each possible combination of seed communities and selects the one with the maximum diversity score.

Measures: To evaluate the efficiency of our Top$L$-ICDE approach, we report the *wall clock time*, which is the time cost to online retrieve Top$L$-ICDE answers via the index (Algorithm 3). For DTop$L$-ICDE, we report the *wall clock time* and *accuracy* (defined as the ratio of the diversity score of our method to that of the optimal method).

Parameters Settings: Table III depicts the parameter settings, where default values are in bold. Each time we vary the values of one parameter, while other parameters are set to their default values. We ran all the experiments on the machines with Intel(R) Core(TM) i9-10900K 3.70GHz CPU, Ubuntu 20.04 OS, and 32 GB memory. All algorithms were implemented in Python and executed with Python 3.11 interpreter.

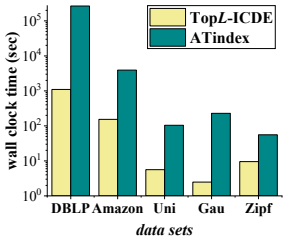
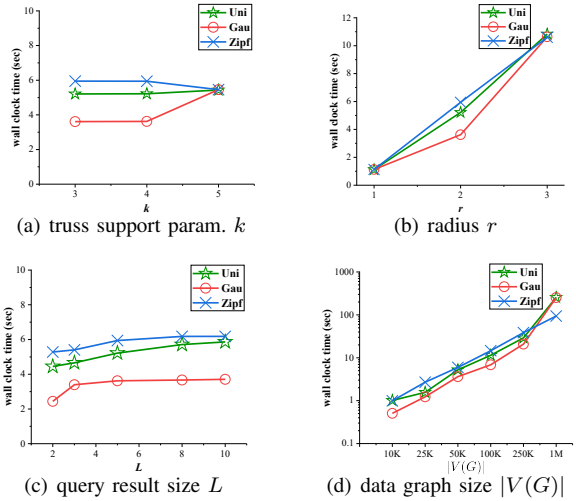


Fig. 2. Top$L$-ICDE performance on real/synthetic graph data.



(a) truss support param. k (b) radius r

(c) query result size L (d) data graph size $|V(G)|$

Fig. 3. The robustness evaluation of the Top$L$-ICDE performance.

Research Questions: We conduct extensive experiments to evaluate our Top$L$-ICDE and DTop$L$-ICDE approaches and answer the following four research questions (RQs):

RQ1 (Efficiency): Can our proposed approaches efficiently process Top$L$-ICDE and DTop$L$-ICDE queries?

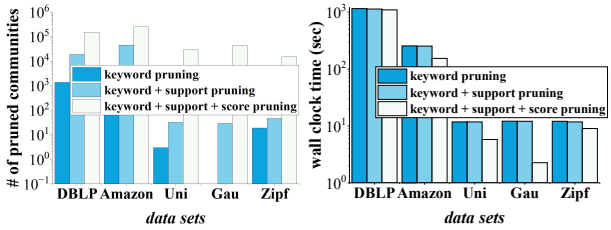
RQ2 (Effectiveness): Can our proposed pruning strategies effectively filter out candidate communities during Top$L$-ICDE query processing?

RQ3 (Meaningfulness): Are the resulting Top$L$-ICDE communities useful for real-world applications?

RQ4 (Accuracy): Can our proposed approach achieve high accuracy of DTop$L$-ICDE query answers?

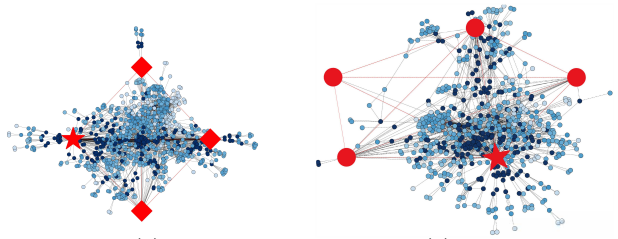
B. Top$L$-ICDE Performance Evaluation

The Top$L$-ICDE Efficiency (RQ1): Figure 2 compares the performance of our Top$L$-ICDE approach with that of *ATindex* over real and synthetic graphs, in terms of the *wall clock time*, where we set all parameters to their default values in Table III. Note that, for *DBLP*, since the time cost of *ATindex* is extremely high, we sample 0.5% center vertices from original graph data without replacement and estimate the total time as $\frac{\overline{t_s}}{0.005} = 200 \cdot \overline{t_s}$, where $\overline{t_s}$ is the average time per sample. The experimental results show that our Top$L$-ICDE approach outperforms *ATindex* by more than one order of magnitude, which confirms the efficiency of our Top$L$-ICDE algorithm on real/synthetic graphs.



(a) pruning power (b) time cost

Fig. 4. The ablation study of the Top$L$-ICDE performance.



(a) Top$L$-ICDE ($\sigma(g) = 344.31$, 974 possibly influenced nodes) (b) k-core ($\sigma(g) = 239.81$, 646 possibly influenced nodes)

Fig. 5. The influenced communities from Top$L$-ICDE vs. k-core ($k = 4$).

To evaluate the robustness of our Top$L$-ICDE approach, in subsequent experiments, we will vary different parameters (e.g., k, r, L, and $|V(G)|$) on synthetic graphs. Due to space limitations, for the effect of other parameters (e.g., θ,$|Q|$,$|v_i.W|$, and $|\Sigma|$), please refer to our technical report [17].

Effect of Truss Support Parameter k: Figure 3(a) illustrates the performance of our Top$L$-ICDE approach, where the support parameter of the truss $k = 3$, 4, and 5, and default values are used for other parameters. The time cost is generally not very sensitive to k values, since edge supports are similar in all the three synthetic graphs. When $k = 5$, however, no candidate communities can be detected, and thus time costs on the three graphs are similar (but trends are different from cases of $k = 3, 4$). For different k values, wall clock times of our Top$L$-ICDE approach remain low (i.e., 3.61 $\sim$ 5.95 sec).

Effect of Radius r: Figure 3(b) illustrates the experimental results of our Top$L$-ICDE approach for different radii r of seed communities, where $r = 1$, 2, and 3 and other parameters are by default. A larger radius r leads to larger seed communities to filter and refine, which incurs higher time costs, as confirmed by the figure. Nonetheless, the time cost remains small (i.e., 1.12 $\sim$ 10.83 sec) for different r values.

Effect of the Size, L, of Query Result Set: Figure 3(c) presents the performance of our Top$L$-ICDE approach with different sizes, L, of query result set, where L varies from 2 to 10 and default values are used for other parameters. Intuitively, the larger L, the more communities must be processed. Despite that, the time cost of Top$L$-ICDE remains low (i.e., 2.44 $\sim$ 6.18 sec) for different L values.

Effect of the Graph Size $|V(G)|$: Figure 3(d) tests the scalability of our Top$L$-ICDE approach with different social network sizes, $|V(G)|$, from $10K$ to $1M$, where default values are assigned to other parameters. In the figure, when the graph size $|V(G)|$ becomes larger, the wall clock time smoothly increases (i.e., from 0.51 sec to 255.62 sec for $|V(G)|$ from $10K$ to $1M$, respectively), which confirms the scalability of our Top$L$-ICDE algorithm for large network sizes.

Ablation Study (RQ2): We conduct an ablation study over real/synthetic graphs to evaluate the effectiveness of our proposed pruning strategies, where all parameters are set to their default values. We tested different combinations by adding one more pruning method each time: (1) *keyword pruning* only, (2) *keyword + support pruning*, and (3) *keyword + support + score pruning*. Figure 4(a) examines the number of pruned candidate communities, whereas Figure 4(b) shows the time cost for different pruning combinations. From experimental results, we can see that with more pruning methods, the number of pruned communities increases by about an order of magnitude, and the wall clock time decreases. Especially, the third influential score pruning method can significantly prune more candidate communities (in addition to the first two pruning methods) and result in the lowest time cost.

Case Study (RQ3): To evaluate the usefulness of our Top$L$-ICDE results, we conduct a case study to compare the influences of our Top$L$-ICDE seed community with that of k-core [24] over $Amazon$. Figure 5(a) shows our Top1-ICDE community with 4 users ($(4, 2)$-truss), whereas Figure 5(b) illustrates 5 users in 4-core community, where the (red) star point in both subfigures represent the same center vertex. The figures show that our Top1-ICDE community has an influential score $\sigma(g) = 344.31$ with 974 possibly influenced users (blue points). In contrast, the 4-core has more seed users, but with a lower influential score $\sigma(g) = 239.81$ and a smaller number of possibly influenced users (i.e., 646). This confirms the usefulness of our Top$L$-ICDE problem to obtain seed communities with high influences for real-world applications such as online advertising/marketing.

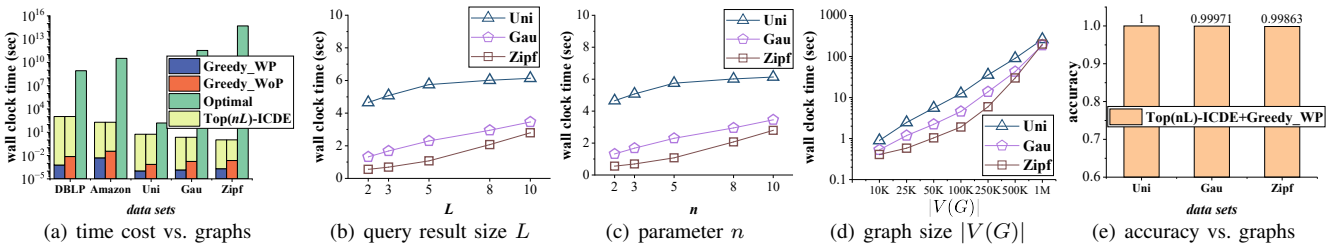
C. DTopL-ICDE Performance Evaluation

The DTop$L$-ICDE Efficiency (RQ1): Figure 6(a) compares the performance of our DTop$L$-ICDE approach (i.e., Top(nL)-ICDE+Greedy_WP), Top(nL)-ICDE+Greedy_WoP, and the *Optimal* algorithm over real and synthetic graphs, in terms of the *wall clock time*, where all parameters are set to their default values in Table III. We can find that DTop$L$-ICDE outperforms *Optimal* by at least three orders of magnitude.

Below, we evaluate the robustness of DTop$L$-ICDE with different parameters (e.g., n, L, $|V(G)|$) on synthetic graphs.

Effect of the Size, L, of Query Result Set: Figure 6(b) shows the experimental results of our DTop$L$-ICDE approach for different sizes, L, of query result set, by varying L from 2 to 10 and default values are used for other parameters. Larger L values lead to lower influential score bound $\sigma_{(nL)}$, and thus more candidate communities to be retrieved and refined, which incur higher time costs. For various L values, the time cost of DTop$L$-ICDE remains low (i.e., 2.72 $\sim$ 6.39 sec).

Effect of Parameter n: Figure 6(c) shows the performance of our DTop$L$-ICDE approach, where n varies from 2 to 10 and other parameters are set to their default values. With increasing n, lower influential score bound $\sigma_{(nL)}$ is used,



(a) time cost vs. graphs (b) query result size L (c) parameter n (d) graph size $|V(G)|$ (e) accuracy vs. graphs

Fig. 6. The DTop$L$-ICDE performance evaluation.

resulting in higher time costs. Nevertheless, the *wall clock time* remains small (i.e., 2.72 $\sim$ 6.28 sec) for various n values.

Effect of the Graph Size $|V(G)|$: Figure 6(d) reports the performance of our DTop$L$-ICDE approach with different social-network sizes, $|V(G)|$, from $10K$ to $1M$ and default values are used for other parameters. Intuitively, the larger $|V(G)|$, the more communities that must be processed, which incurs smoothly increasing time costs (i.e., 0.9 $\sim$ 278.18 sec).

The DTop$L$-ICDE Accuracy (RQ4): We test the experiments on small-scale graphs ($|V(G)| = 1K$, 3 keywords per vertices, and $|\Sigma| = 20$) following Uniform, Gaussian, and Zipf distributions, and report the *accuracy* of our DTop$L$-ICDE approach (i.e., the diversity score ratio of our approach to *Optimal*) in Figure 6(e). The results indicate that our DTop$L$-ICDE accuracy is very close to 100% (i.e., 99.863% $\sim$ 100%).

IX. Related Work

Community Search/Detection: Prior works proposed many community semantics based on different structural cohesiveness, such as the minimum degree [25], k-core [24], k-clique [26], and (k, d)-truss [12], [23]. In contrast, our Top$L$-ICDE problem retrieves not only highly connected seed communities but also those with the highest influences and containing query keywords in social networks, which is more challenging. On the other hand, previous works on community detection retrieved all communities by considering link information only [27], [28]. More recent works used clustering techniques to detect communities [29]–[31]. However, these works did not require structural constraints of community answers or consider the impact of the influenced communities, which is the focus of our Top$L$-ICDE problem.

Influence Maximization: Previous works on the *influence maximization* (IM) problem [14] over social networks [13], [32]–[35] usually obtain arbitrary individual users from social networks with the maximum influence on other users, where *independent cascade* (IC) and *linear threshold* (LT) models [14] were used to capture influence propagation. However, most solutions to the IM problem do not assume strong social relationships among selected seed users. In contrast, our Top$L$-ICDE requires seed communities to be connected, have high structural cohesiveness, and cover some query keywords, which is more challenging.

Influential Community: There are some recent works [19], [36], [37] on finding the most influential community over social networks. These works considered different graph data models such as uncertain graphs [19] and heterogeneous information networks [37] and influential community semantics like kr-clique [36], (k, η)-influential community, and $(k, \mathcal{P})$-core [37]. Moreover, they ignored the interests of users (represented by keywords) in communities. With different graph models and influential community semantics, our Top$L$-ICDE problem uses a certain, undirected graph data model with the MIA model for the influence propagation and aims to retrieve top-L influential communities (rather than all communities) under different community semantics of structural, keyword-aware, and influential score ranking.

Diversified Subgraphs: There are several existing works that consider retrieving the diversified subgraphs. For example, Yang et al. [38] studied the top-k diversified subgraph problem, which returns a set of up to k subgraphs that are isomorphic to a given query graph, and cover the largest number of vertices. Some prior works [39], [40] studied the structural diversity search problem in graphs, which obtains vertex(es) with the highest structural diversities (defined as # of connected components in the 1-hop subgraph of a vertex). Chowdhary et al. [41] aimed to search for a community that is structure-cohesive (i.e., with the minimum number of vertices) and attribute-diversified (i.e., with the maximum number of attribute labels in vertices). The aforementioned works either did not consider the cohesiveness and/or influences of the returned subgraphs, or focused on node-/attribute-level diversity (rather than community-level diversity). Thus, with different problem definitions, we cannot directly use techniques proposed in these works to solve our DTop$L$-ICDE problem.

X. Conclusions

In this paper, we propose a novel Top$L$-ICDE problem, which retrieves top-L communities from social networks with the highest influential scores.We provide effective pruning strategies to rule out false alarms of candidate communities and design an index to facilitate an efficient Top$L$-ICDE processing algorithm. We also formulate and tackle an NP-hard Top$L$-ICDE variant, DTop$L$-ICDE, by proposing a greedy algorithm with effective pruning strategies. Experimental results on real/synthetic graphs confirm the good performance of our proposed Top$L$-ICDE and DTop$L$-ICDE approaches.

Acknowledgements

This work was supported by Natural Science Foundation of China (62272170), Natural Science Foundation (NSF CCF-2217104), and Shanghai International Joint Lab of Trustworthy Intelligent Software (22510750100). Mingsong Chen is the corresponding author.

REFERENCES

[1] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2010, pp. 1029–1038.

[2] Y. Tang, Y. Shi, and X. Xiao, "Influence maximization in near-linear time: A martingale approach," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, 2015, pp. 1539–1554.

[3] S. Tu and S. Neumann, "A viral marketing-based model for opinion dynamics in online social networks," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 1570–1578.

[4] X. Song, J. Lian, H. Huang, M. Wu, H. Jin, and X. Xie, "Friend recommendations with self-rescaling graph neural networks," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 3909–3919.

[5] J. Fan, J. Qiu, Y. Li, Q. Meng, D. Zhang, G. Li, K.-L. Tan, and X. Du, "OCTOPUS: An Online Topic-Aware Influence Analysis System for Social Networks," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, Apr. 2018, pp. 1569–1572.

[6] K. Wang, S. Wang, X. Cao, and L. Qin, "Efficient Radius-Bounded Community Search in Geo-Social Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4186–4200, Sep. 2022.

[7] Q. Liu, M. Zhao, X. Huang, J. Xu, and Y. Gao, "Truss-based Community Search over Large Directed Graphs," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, May 2020, pp. 2183–2197.

[8] L. Sun, X. Huang, R.-H. Li, B. Choi, and J. Xu, "Index-based intimate-core community search in large weighted graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 9, pp. 4313–4327, 2020.

[9] B. Liu, F. Zhang, W. Zhang, X. Lin, and Y. Zhang, "Efficient community search with size constraint," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 97–108.

[10] C.-Y. Wang, Y. Chen, and K. R. Liu, "Game-theoretic cross social media analytic: How yelp ratings affect deal selection on groupon?" *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 5, pp. 908–921, 2017.

[11] X. Huang and L. V. S. Lakshmanan, "Attribute-Driven Community Search," *VLDB-2017*, vol. 10, no. 9, pp. 949–960, 2017.

[12] A. Al-Baghdadi and X. Lian, "Topic-based community search over spatial-social networks," *Proceedings of the VLDB Endowment*, vol. 13, no. 12, pp. 2104–2117, Aug. 2020.

[13] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," in *Proceedings of the International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, ser. KDD '10, Jul. 2010, pp. 1029–1038.

[14] D. Kempe, J. Kleinberg, and É. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2003, pp. 137–146.

[15] U. Feige, "A threshold of ln n for approximating set cover," *Journal of the ACM*, vol. 45, no. 4, pp. 634–652, Jul. 1998.

[16] J. Cohen, "Trusses: Cohesive subgraphs for social network analysis," *National security agency technical report*, vol. 16, no. 3.1, 2008.

[17] N. Zhang, Y. Ye, X. Lian, and M. Chen, "Top-L Most Influential Community Detection Over Social Networks (Technical Report)," Nov. 2023. [Online]. Available: https://arxiv.org/abs/2311.13162

[18] U. Feige, "A threshold of ln n for approximating set cover (preliminary version)," in *STOC-96*, ser. STOC '96, Jul. 1996, pp. 314–318.

[19] W. Luo, X. Zhou, K. Li, Y. Gao, and K. Li, "Efficient Influential Community Search in Large Uncertain Graphs," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 4, pp. 3779–3793, Apr. 2023.

[20] Y. Zhou, Y. Fang, W. Luo, and Y. Ye, "Influential Community Search over Large Heterogeneous Information Networks," *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 2047–2060, Apr. 2023.

[21] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang, "Online topic-aware influence maximization," *Proceedings of the VLDB Endowment*, vol. 8, no. 6, pp. 666–677, Feb. 2015.

[22] M. E. J. Newman and D. J. Watts, "Renormalization group analysis of the small-world network model," *Physics Letters A*, vol. 263, no. 4, pp. 341–346, Dec. 1999.

[23] X. Huang and L. V. S. Lakshmanan, "Attribute-driven community search," *Proceedings of the VLDB Endowment*, vol. 10, no. 9, pp. 949–960, May 2017.

[24] M. Sozio and A. Gionis, "The community-search problem and how to plan a successful cocktail party," in *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '10, Jul. 2010, pp. 939–948.

[25] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '14, Jun. 2014, pp. 991–1002.

[26] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang, "Online search of overlapping communities," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13, Jun. 2013, pp. 277–288.

[27] M. E. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, p. 026113, 2004.

[28] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[29] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng, "A model-based approach to attributed graph clustering," in *Proceedings of International Conference on Management of Data (SIGMOD)*, 2012, pp. 505–516.

[30] A. Conte, T. De Matteis, D. De Sensi, R. Grossi, A. Marino, and L. Versari, "D2k: scalable community detection in massive networks via small-diameter k-plexes," in *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2018, pp. 1272–1281.

[31] N. Veldt, D. F. Gleich, and A. Wirth, "A Correlation Clustering Framework for Community Detection," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, Apr. 2018, pp. 439–448.

[32] J. Tang, X. Tang, X. Xiao, and J. Yuan, "Online processing algorithms for influence maximization," in *Proceedings of the 2018 International Conference on Management of Data*, 2018, pp. 991–1005.

[33] N. Ohsaka, "The solution distribution of influence maximization: A high-level experimental study on three algorithmic approaches," in *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, 2020, pp. 2151–2166.

[34] J. Ali, M. Babaei, A. Chakraborty, B. Mirzasoleiman, K. P. Gummadi, and A. Singla, "On the Fairness of Time-Critical Influence Maximization in Social Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2875–2886, Mar. 2023.

[35] D. Li, J. Liu, J. Jeon, S. Hong, T. Le, D. Lee, and N. Park, "Large-scale data-driven airline market influence maximization," in *Proceedings of ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2021, pp. 914–924.

[36] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu, "Most influential community search over large social networks," in *2017 IEEE 33rd international conference on data engineering (ICDE)*. IEEE, 2017, pp. 871–882.

[37] Y. Zhou, Y. Fang, W. Luo, and Y. Ye, "Influential Community Search over Large Heterogeneous Information Networks," *Proceedings of the VLDB Endowment*, vol. 16, no. 8, pp. 2047–2060, Apr. 2023.

[38] Z. Yang, A. W.-C. Fu, and R. Liu, "Diversified Top-k Subgraph Querying in a Large Graph," in *Proceedings of the International Conference on Management of Data (SIGMOD)*, ser. SIGMOD '16, Jun. 2016, pp. 1167–1182.

[39] X. Huang, H. Cheng, R.-H. Li, L. Qin, and J. X. Yu, "Top-K structural diversity search in large networks," *The VLDB Journal*, vol. 24, no. 3, pp. 319–343, Jun. 2015.

[40] C.-H. Tai, P. S. Yu, D.-N. Yang, and M.-S. Chen, "Structural Diversity for Resisting Community Identification in Published Social Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 1, pp. 235–252, Jan. 2014.

[41] A. A. Chowdhary, C. Liu, L. Chen, R. Zhou, and Y. Yang, "Finding attribute diversified community over large attributed networks," *World Wide Web*, vol. 25, no. 2, pp. 569–607, Mar. 2022.